Digital Signal Processing
Assignment 2 – FIR filters

Form groups of 3-4 people. You can find team mates on the moodle page.

The task of this assignment is to filter an ECG with FIR filters and to detect the 'R' peaks. In contrast to the FFT assignment, we write filter code which can be used for realtime processing. This means that the FIR filter needs to be implemented with the help of delay lines and the impulse response is truncated.

ECG filtering

A unique ECG pattern will be allocated to each group.

1. Create two functions which calculate and return the FIR filter coefficients numerically (using python's IFFT command) for:
   (a) a highpass filter and
   (b) a bandstop filter
   Name these functions "highpassDesign" and "bandstopDesign". Both functions should automatically decide how many coefficients are required. The function arguments should be the sampling rate and the cutoff frequencies (and any other optional arguments you'd like to provide). Feel free to put both functions in a class. **[25%]**

2. Create a Python FIR filter class which implements an FIR filter which has a method of the form value dofilter(self, value) where both the value argument and return value are *scalars* and not vectors so that it can be used in a realtime system! The constructor of the class takes the coefficients as its input:
   ```
   Class FIRfilter:
   def __init__(self, _coefficients):
   # your code here
   def dofilter(self, v):
   # your code here
   return result
   ```
   Filter your ECG with the above FIR filter class using the coefficients from Q1 by removing the 50 Hz interference and the baseline wander with the highpass. Decide which cutoff frequencies are needed and provide explanations by referring to the spectra and/or fundamental frequencies of the ECG. Simulate realtime processing by feeding the ECG sample by

sample into your FIR filter class. Make sure that the ECG looks intact and that it is not distorted (PQRST intact). Provide appropriate plots. **[25%]**

3. Instead of using a bandstop filter to remove the 50 Hz, use an adaptive LMS filter by providing it with a 50 Hz sine wave as the reference noise. Add an adaptive LMS filter command to your FIR filter class and name it: “doFilterAdaptive(self, signal, noise, learningRate)” which returns the cleaned up ECG. As before this function must receive only scalars (i.e. sample by sample) and return a scalar. Plot and compare the result from this to that from the FIR filter design. **[25%]**

4. ECG heartbeat detection: The task is to detect R-peaks in your ECG. Write a matched filter which either uses a wavelet as your R-peak template (because wavelets look pretty much like R-peaks) or cut out a single ECG trace and use this as the template. If using a wavelet, make sure it has the correct duration and shape that it matches your R-peaks. In the report it is important that you show a plot of a real R peak and the wavelet/template side by side. Plot the momentary heartrate (i.e. inverse intervals between R-peaks) against time. **[25%]**

Each team’s report must be based on a different ECG recording. Please keep the reports short but it should make clear what you have done and why you have done it. Include the complete Python code as well as the plots of the ECGs (time domain) and their frequency representation (with proper labels). If necessary, annotate the plots with InkScape, Corel Draw or Illustrator by labelling the ECG peaks and remember to use vector-based image formats, for example EPS, SVG, PDF or EMF and not pixel-based formats for the report. These figures need to be in the report at the correct place and not attached separately. Also, show zoomed in ECG traces of one heartbeat so that it is possible to identify the different parts of a single heartbeat and that it’s still possible to check if it is still intact.

No high level Python functions except FFT/IFFT and the window functions are allowed. Any use of “lfilter”, “firwin”, “conv”, “correl” and any a-causal processing (i.e. data array in and array out) commands will result in zero or very low marks. As before, submit a zip file containing all files and test the zip before submission by unzipping it and then running the python from the command line in a terminal (not spyder). Also check that all plots are generated when running the scripts from the command line. See moodle for the exact filename conventions and submission details.

Deadline: 15th November, 5pm.